PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-022599
(43)Date of publication of application : 26.01.2001

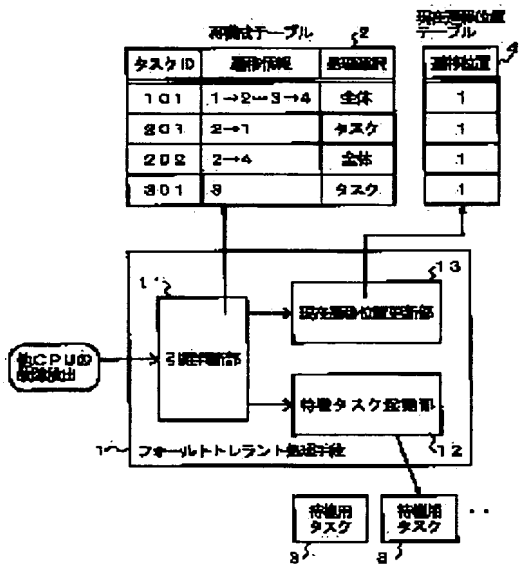
(51)Int.Cl. G06F 9/46
G06F 11/20
G06F 15/177

(21)Application number : 11-191135
(22)Date of filing : 06.07.1999

(71)Applicant : FUJITSU LTD
(72)Inventor : MATSUSHITA KOZO
MINAZU SHINJI
OKAMOTO JIRO

(54) FAULT TOLERANT SYSTEM, FAULT TOLERANT PROCESSING METHOD AND RECORDING MEDIUM FOR FAULT TOLERANT CONTROL PROGRAM

(57)Abstract:
PROBLEM TO BE SOLVED: To easily designate a specific CPU that takes over the processing of another faulty CPU for every task in a multiprocessor system.
SOLUTION: A take-over decision part 11 searches for a task which is operating with a faulty CPU from the transition information, i.e., the series of the CPU number that takes over the task included in a reconfiguration table 2 and the current transition position showing the CPU under processing in a current transition position table 4 after the CPU fault is detected and then checks the CPU that takes over the task. When the checked CPU is its own one, a standby task start part 12 starts a standby task 3 of the relevant task to take over the processing. A current transition position updating part 13 updates the current transition position of the table 4. When no CPU takes over the task, the task or a total system is discontinued according to the processing selection information.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－22599
（P2001－22599A）
(43)公開日　平成13年１月26日(2001.1.26)

(51)Int.Cl.'		識別記号	FI		テーマコード'（参考）
Ｇ０６Ｆ	9/46	３６０	Ｇ０６Ｆ	9/46	３６０Ｂ　５Ｂ０３４
		３３０			３３０Ｃ　５Ｂ０４５
	11/20	３１０		11/20	３１０Ｆ　５Ｂ０９８
	15/177	６７８		15/177	６７８Ａ

審査請求　未請求　請求項の数６　ＯＬ　（全 12 頁）

(21)出願番号　特願平11－191135

(22)出願日　平成11年７月６日(1999.7.6)

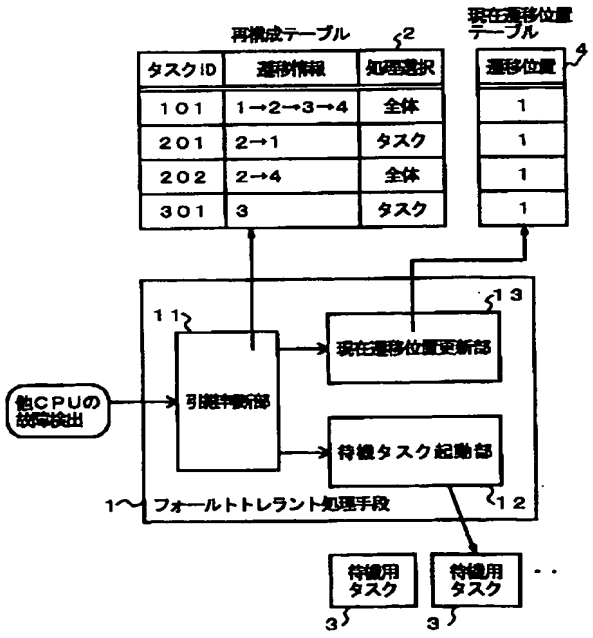
(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(72)発明者　松下　耕三
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　水津　真治
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(74)代理人　100087848
弁理士　小笠原　吉義　（外２名）

最終頁に続く

(54)【発明の名称】　フォールトトレラント・システム，フォールトトレラント処理方法およびフォールトトレラント制御用プログラム記録媒体

(57)【要約】
【課題】　マルチプロセッサシステムにおいて，あるＣＰＵが故障した場合に，そのＣＰＵの処理を他のＣＰＵがタスク単位で引き継ぐときに，どのＣＰＵが引き継ぐかを簡単に指定することができるようにすることを目的とする。
【解決手段】　引継判断部11は，他のＣＰＵの故障を検出すると，再構成テーブル2 中のタスクを引き継ぐＣＰＵ番号の系列である遷移情報と，現在遷移位置テーブル4 中の現在処理しているＣＰＵを示す現在遷移位置から，故障したＣＰＵで稼働していたタスクを探し，そのタスクを引き継ぐ次のＣＰＵを調べる。引き継ぐＣＰＵが自ＣＰＵであれば，待機タスク起動部12は，そのタスクの待機用タスク3 を起動し，処理の引き継ぎを行う。現在遷移位置更新部13は，現在遷移位置テーブル4 の現在遷移位置を更新する。タスクを引き継ぐＣＰＵがない場合，処理選択情報に基づきタスクまたはシステム全体を停止させる。

本発明のブロック構成図



1

【特許請求の範囲】

【請求項１】　プロセッサの実行処理単位である複数のタスクが複数のプロセッサに分散して配置され，各プロセッサにおいて処理されるマルチプロセッサシステムにおいて，タスクごとに，プロセッサの故障時にその故障プロセッサで動作していたタスクを引き継ぐプロセッサの系列を定義した遷移情報と，タスクが現在どのプロセッサ上で動作しているのかを示す現在遷移位置とを記憶する記憶手段と，あるプロセッサが故障した場合に，前記遷移情報および前記現在遷移位置の情報にもとづいて，その故障プロセッサで動作していたタスクが自プロセッサで引き継ぐべきタスクであるかどうかを判断する手段と，判断の結果，自プロセッサで引き継ぐものである場合には，そのタスクの引き継ぎを行い，自プロセッサで引き継いだタスクを動作させる手段とを備えることを特徴とするフォールトトレラント・システム。

【請求項２】　前記記憶手段は，タスクを引き継ぐプロセッサがない場合に，そのタスクの停止またはシステム全体の停止のいずれかを示す処理選択情報を記憶し，前記プロセッサは，故障したプロセッサで動作していたタスクを引き継ぐプロセッサがない場合に，前記処理選択情報にもとづいてタスクまたはシステム全体を停止させることを特徴とする請求項１記載のフォールトトレラント・システム。

【請求項３】　プロセッサの実行処理単位である複数のタスクが複数のプロセッサに分散して配置され，各プロセッサにおいて処理されるマルチプロセッサシステムにおけるフォールトトレラント処理方法において，タスクごとに，プロセッサの故障時にその故障プロセッサで動作していたタスクを引き継ぐプロセッサの系列を定義した遷移情報と，タスクが現在どのプロセッサ上で動作しているのかを示す現在遷移位置とを記憶する記憶手段を用い，あるプロセッサが故障した場合に，前記記憶手段を参照し，前記遷移情報および前記現在遷移位置の情報にもとづいて，その故障プロセッサで動作していたタスクが自プロセッサで引き継ぐべきタスクであるかどうかを判断し，判断した結果，自プロセッサで引き継ぐものである場合には，そのタスクの引き継ぎを行い，自プロセッサで引き継いだタスクを動作させることを特徴とするフォールトトレラント処理方法。

【請求項４】　タスクを引き継ぐプロセッサがない場合に，そのタスクの停止またはシステム全体の停止のいずれかを示す処理選択情報を前記記憶手段に記憶しておき，故障したプロセッサで動作していたタスクを引き継ぐプロセッサがない場合に，前記処理選択情報にもとづいてタスクまたはシステム全体を停止させることを特徴とする請求項３記載のフォールトトレラント処理方法。

【請求項５】　マルチプロセッサシステムにおけるフォールトトレラント処理方法をコンピュータによって実現するためのプログラムを記録した記録媒体であって，タスクごとに，プロセッサの故障時にその故障プロセッサで動作していたタスクを引き継ぐプロセッサの系列を定義した遷移情報と，タスクが現在どのプロセッサ上で動作しているのかを示す現在遷移位置とを記憶する記憶手段を用い，あるプロセッサが故障した場合に，前記記憶手段を参照し，前記遷移情報および前記現在遷移位置の情報にもとづいて，その故障プロセッサで動作していたタスクが自プロセッサで引き継ぐべきタスクであるかどうかを判断し，判断した結果，自プロセッサで引き継ぐものである場合には，そのタスクの引き継ぎを行い，自プロセッサで引き継いだタスクを動作させる処理を，コンピュータに実行させるプログラムを記録したことを特徴とするフォールトトレラント制御用プログラム記録媒体。

【請求項６】　請求項５記載のフォールトトレラント制御用プログラム記録媒体において，前記プログラムは，前記記憶手段に前記遷移情報とともに記憶された処理選択情報であって，タスクの停止またはシステム全体の停止のいずれかを示す処理選択情報にもとづいて，故障したプロセッサで動作していたタスクを引き継ぐプロセッサがない場合に，タスクまたはシステム全体を停止させる処理を，コンピュータに実行させるプログラムを含むことを特徴とするフォールトトレラント制御用プログラム記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は，複数のプロセッサ（以下，ＣＰＵという）を搭載したシステムにおけるフォールトトレラント技術に関し，特に，あるＣＰＵが故障した場合に，そのＣＰＵが行っていた処理を他の正常なＣＰＵがタスク単位で引き継ぐとき，どのＣＰＵが引き継ぐのかを正常なＣＰＵが最後の一つになるまで，容易に指定できるようにしたマルチＣＰＵシステムにおけるフォールトトレラント・システム，フォールトトレラント処理方法およびフォールトトレラント制御用プログラム記録媒体に関する。

【０００２】

【従来の技術】マルチＣＰＵシステムにおいて，あるＣＰＵが故障したときに，システムを停止することなく，他の正常なＣＰＵによってその処理を継続できるようにするため，例えば，以下のようなフォールトトレラント技術がある。

【０００３】（１）「相互ホットスタンバイシステム待機系選択方式（特開平９－８１４０９号公報）」は，電子計算機の障害により稼働系処理機能に対応付けられた待機系処理機能が存在しなくなったとき，操作員の介入なしに自動的に対応付け可能な待機系処理機能を選択して対応付け，ホットスタンバイ関係を構築するためのものであり，自電子計算機で動作する稼働系処理機能の識別子，稼働系処理機能とホットスタンバイ関係を構築す

るように対応付けられた待機系処理機能の識別子および該待機系処理機能が動作する電子計算機の番号からなる情報と，自電子計算機で動作する待機系処理機能の識別子，待機系処理機能とホットスタンバイ関係を構築するように対応付けられた稼働系処理機能の識別子および該稼働系処理機能が動作する電子計算機の番号からなる情報とを登録する処理機能管理表を備えるとともに，電子計算機の障害発生時に，対応付けられた待機系処理機能がなくなった稼働系処理機能からの要求に応じて，前記処理機能管理表を参照して他の電子計算機の対応付けられた稼働系処理機能のない待機系処理機能を選択して要求元の稼働系処理機能に対応付け，自動的にホットスタンバイ関係を構築させる処理機能管理手段とを備える。

【０００４】（２）また，「業務引継システム（特願平９－５２８７９４号）」は，ホットスタンバイの形態やロードシェアの形態で，複数の処理装置により業務処理の実行を行うシステムにおいて，特に多大なプログラミングを必要とせずに障害発生時の効率的な引き継ぎができるような業務引継システムを提供することを目的として，各処理装置が業務についての現用系であるか待機系であるかを表すテーブルを保持し，業務についての現用系である処理装置の障害発生時に，該障害が発生した処理装置における業務に係る処理を，前記テーブルを参照して当該業務の待機系となる処理装置に引き継ぐようにするものである。

【０００５】

【発明が解決しようとする課題】しかし，従来，多数のＣＰＵが次々に故障し続けたとき，故障が生じたＣＰＵの処理をどのＣＰＵが引き継ぐのかを容易に指定する方法はなかった。

【０００６】また，上記（１）の技術では，処理機能管理表により，稼働系処理識別子と待機系識別子を管理する必要がある。また，この技術では，処理を引き継ぐ計算機が動的に定まるので，事前に処理を引き継ぐＣＰＵを最後の１台まで静的に定めることができず，さらに，すべてのＣＰＵが同等なハードウェアなどの機能を有している必要があるという問題があった。

【０００７】また，上記（２）の技術では，全クラスタのテーブルを作成する必要があり，全クラスタが故障するまで引き継がせたい業務があれば，すべてのテーブルに待機業務を記さなければならない。また，処理を引き継ぐクラスタが待機系を用意している中のどのクラスタになるかは不明であり，優先度を付けてクラスタに処理を引き継がせるような設定をすることができないという問題があった。

【０００８】本発明は上記問題点の解決を図り，マルチＣＰＵシステムにおいて，あるＣＰＵが故障した場合に，そのＣＰＵの処理をタスク単位で引き継ぐとき，どのＣＰＵが引き継ぐのかを正常なＣＰＵが最後の１台になるまで，容易に指定することができる手段を提供することを目的とする。

【０００９】

【課題を解決するための手段】本発明は，複数のＣＰＵを搭載したシステムにおいて，各ＣＰＵは，故障したＣＰＵで動作していたタスクをどのＣＰＵによって引き継ぐのかを，タスクごとにＣＰＵ番号の系列によって記した情報を持つ再構成テーブルを持つ。故障したＣＰＵを発見した場合，それを発見したＣＰＵは，他の全ＣＰＵに対し故障したＣＰＵのＣＰＵ番号（各ＣＰＵを識別するための識別番号）を知らせる。各ＣＰＵは，再構成テーブルをもとに，故障したＣＰＵ上で動作していたタスクを知り，そのタスクを引き継ぐＣＰＵが自分であるかどうかを判断する。自ＣＰＵが引き継ぐように指定されているタスクであった場合，そのタスクを引き継ぐために待機していたタスクを起動させる。

【００１０】この引き継ぎのため，再構成テーブルには，タスクがどのＣＰＵによって引き継がれていくのかをＣＰＵ番号の羅列で表現するだけでよいので，容易に引き継いでいくＣＰＵを指定することができる。また，現在どのＣＰＵ上でタスクが実行されているのかを覚えておくことにより，簡単に引き継ぐＣＰＵがどのＣＰＵであるかを調べることができる。

【００１１】また，再構成テーブルには，引き継ぐＣＰＵが存在しなくなった場合に，このタスクだけを停止させるのか，またはシステム全体として停止させるのかを指定する情報を持たせておく。これによって，タスクだけの停止，システム全体の停止を選択することができる。

【００１２】なお，ここでいうＣＰＵ（central processing unit）には，ＭＰＵ（microprocessing unit）も含まれる。また，ここでいうタスクは，ＣＰＵで実行される処理の単位を意味し，プロセスと呼ばれるような処理単位も含まれる広い概念のものである。

【００１３】以上の各処理手段をコンピュータによって実現するためのプログラムは，コンピュータが読み取り可能な可搬媒体メモリ，半導体メモリ，ハードディスクなどの適当な記録媒体に格納することができる。

【００１４】

【発明の実施の形態】図１は，本発明の概要を示すブロック構成図である。本実施の形態では，マルチＣＰＵシステムにおいて，ＣＰＵごとに，フォールトトレラント処理手段１および再構成テーブル２を備える。また，どのＣＰＵもアクセスすることができる記憶領域に，現在遷移位置テーブル４を備える。

【００１５】再構成テーブル２は，タスクごとに，各タスクを識別するタスクＩＤ（識別子）と，故障が生じたＣＰＵ上で実行されていたタスクを，どのＣＰＵによって引き継ぐのかをＣＰＵ番号（各ＣＰＵを識別するためのシーケンシャルな番号）の系列で指定する遷移情報と，引き継ぐＣＰＵが存在しない場合に，そのタスクだ

けを停止するか，システム全体を停止するかを指定する処理選択情報を持つ。

【００１６】現在遷移位置テーブル４は，再構成テーブル２に記憶されている各タスクが現在それぞれどのＣＰＵで稼働しているかを，再構成テーブル２における遷移情報の系列の位置で示す現在遷移位置情報を保持する。

【００１７】フォールトトレラント処理手段１は，他ＣＰＵに故障が生じた場合に，再構成テーブル２の遷移情報をもとに，その故障したＣＰＵ上で実行中のタスクが，自分（自ＣＰＵ）が引き継ぐものかどうかを判断する引継判断部１１と，そのタスクの処理を自ＣＰＵが引き継ぐ場合に，そのタスクの待機用タスク３を起動して，引き継ぎを行う待機タスク起動部１２と，再構成テーブル２の現在遷移位置を自ＣＰＵを示す位置に更新する現在遷移位置更新部１３とを備える。待機用タスク３は，あらかじめ生成しておいてもよく，また必要になったときに新たに生成して，制御を渡すことにより起動してもよい。

【００１８】図２は，本発明の処理動作の流れの例を示す。この例では，ＣＰＵ１～ＣＰＵ４の番号を付与された４つのＣＰＵからなるマルチプロセッサシステムにおいて，タスクＩＤが１０１のタスク（以下，タスク１０１と表記する），タスクＩＤが２０１のタスク（以下，タスク２０１と表記する），タスクＩＤが２０２のタスク（以下，タスク２０２と表記する），タスクＩＤが３０１のタスク（以下，タスク３０１と表記する）の４つのタスクが実行されているとする。

【００１９】再構成テーブル２には，あらかじめ各タスクごとに，タスクＩＤと障害発生時にそのタスクを引き継ぐＣＰＵ番号の順序を遷移情報として設定しておく。また，引き継ぐＣＰＵがない場合にタスクだけを停止させるのか，システム全体として停止させるのかを示す処理選択の情報を設定しておく。タスク１０１のタスクを例に説明すると，再構成テーブル２には，それを実行しているＣＰＵの障害時に，ＣＰＵ１→ＣＰＵ２→ＣＰＵ３→ＣＰＵ４の順に処理を引き継ぐことが設定され，引き継ぐＣＰＵがない場合にはシステム全体の処理を停止するように設定されている。

【００２０】現在遷移位置テーブル４には，再構成テーブル２に登録されている各タスクごとに，現在実行しているＣＰＵが遷移情報中の先頭から何番目であるかを示す情報が，現在遷移位置として格納されている。故障しているＣＰＵがない初期状態においては，現在遷移位置はすべて「１」である。すなわち，故障発生前では，タスク１０１はＣＰＵ１上で，タスク２０１，２０２はＣＰＵ２上で，タスク３０１はＣＰＵ３上で，それぞれ実行されていることが示されている。

【００２１】ＣＰＵ２が故障したとすると，それを検出したＣＰＵから，ＣＰＵ１，ＣＰＵ３，ＣＰＵ４は，ＣＰＵ２の故障通知を受信する（ステップＳ１）。それぞれのＣＰＵでは，再構成テーブル２を参照して，ＣＰＵ２上で実行されていたタスク，すなわち遷移情報における１番目の現在遷移位置が「２（ＣＰＵ２）」となっているタスクを探す（ステップＳ２）。ここで，ＣＰＵ２上で実行されていたタスクが，タスク２０１，２０２であることがわかる。

【００２２】ＣＰＵ１は，タスク２０１の遷移情報（２→１）からこのタスクを引き継ぐＣＰＵが自ＣＰＵであることがわかり，ＣＰＵ４は，タスク２０２の遷移情報（２→４）からこのタスクを引き継ぐＣＰＵが自ＣＰＵであることがわかる。そこで，ＣＰＵ１は，２０１用タスクを起動させ，ＣＰＵ４は，２０２用タスクを起動させる（ステップＳ３）。

【００２３】ＣＰＵ１，ＣＰＵ４は，現在遷移位置テーブル４のタスク２０１，２０２の現在遷移位置をそれぞれ「１」から「２」に更新する（ステップＳ４）。

【００２４】その後，ＣＰＵ１，ＣＰＵ３，ＣＰＵ４が稼働している状態で，さらにＣＰＵ１が故障したとする。それを検出したＣＰＵから，ＣＰＵ３，ＣＰＵ４は，ＣＰＵ１の故障通知を受信する（ステップＳ５）。ＣＰＵ３，ＣＰＵ４は，現在遷移位置１，２，２，１をもとに，再構成テーブル２におけるタスク１０１，２０１，２０２，３０１の遷移情報のうち，それぞれ１番目，２番目，２番目，１番目を調べ，ＣＰＵ１で実行されていたタスクがタスク１０１，２０１であることを知る。これらのタスク１０１，２０１の遷移情報から，引き継ぐＣＰＵ番号を得ると，タスク１０１については，引き継ぐべきＣＰＵ２が既に故障しているので，さらに次の遷移情報を調べて，次のＣＰＵ３が引き継ぐべきＣＰＵであることがわかる。一方，タスク２０１は，引き継ぐべきＣＰＵがないことがわかる（ステップＳ６）。

【００２５】そこで，ＣＰＵ３では１０１用タスクを起動させる（ステップＳ７）。引き継ぐべきＣＰＵのないタスク２０１は，再構成テーブル２の処理選択情報を見ると「タスク」であるので，システム全体を停止させるのではなく，タスク２０１だけを停止させる。その後，現在遷移位置テーブル４におけるタスク１０１の現在遷移位置を「３」に更新し，引き継ぐべきＣＰＵのないタスク２０１の現在遷移位置については，「なし」を示す「－１」に更新する（ステップＳ８）。

【００２６】このように，本発明では，故障したＣＰＵ上で稼働していたタスクを引き継ぐＣＰＵの引き継ぎ順序を，再構成テーブル２にＣＰＵの番号を羅列して記すことにより，タスク単位でどのＣＰＵが引き継ぐべきかを正常なＣＰＵが最後の一つになるまで，容易に指定することができる。さらに，現在どのＣＰＵ上でタスクが実行されているのかを示すことで，次に引き継ぐＣＰＵを簡単に調べることができる。また，引き継ぐＣＰＵがなくなった場合の処理も簡単に設定することができ，タスクを停止させるだけで，他のタスクにより処理を続行

させるか，システム全体を停止させるかを容易に指定することができる。

【００２７】このため，ある処理を実行するのに必要な通信回線やディスク記憶装置などのハードウェアを，すべてのＣＰＵに同等に用意する必要がなく，タスクの処理機能に応じて特定のハードウェアが備わっているＣＰＵにのみ確実に引き継がせることができるようになる。

【００２８】図３は，本発明が利用される複数のＣＰＵを搭載したシステムの例を示す。複数のＣＰＵの接続形態としては，一般に知られているように，疎結合型と密結合型とがある。本発明はそのどちらの形態においても適用することができる。

【００２９】図３（Ａ）は，疎結合型の接続形態の例であって，各ＣＰＵ２０－１～２０－ｎは，自己専用のプロセッサバス２１－１～２１－ｎとローカルメモリ２２－１～２２－ｎとを持ち，ＣＰＵ２０－１～２０－ｎ間で共有できるのはシステムバス２３を通したグローバルメモリ２４だけである。

【００３０】図３（Ｂ）は密結合型の接続形態の例であって，各ＣＰＵ３０－１～３０－ｍはプロセッサバス３１もローカルメモリ３２も共有し，ローカルメモリ３２内を各ＣＰＵ３０－１～３０－ｍがそれぞれ専用的に使うメモリ空間とＣＰＵ間で共有的に使うメモリ空間とを論理的に区分けして使用する。

【００３１】図４に，疎結合型のマルチＣＰＵシステムにおけるブロック構成例を示す。ここでは図を簡単に表すために，２台のＣＰＵを示しているが，多数のＣＰＵが接続されていてもよい。本発明に必要な各機能は，ＣＰＵ２０－１，２０－１のローカルメモリ２２－１，２２－２に配置され，また，ＣＰＵ間の共通資源を管理する領域はグローバルメモリ２４に配置される。

【００３２】図５に，密結合型のマルチＣＰＵシステムにおけるブロック構成例を示す。本発明に必要な各手段は，ＣＰＵ３０－１，３０－２ごとに，ローカルメモリ３２の各ＣＰＵ専用メモリ空間に配置され，また，ＣＰＵ間の共通資源を管理する領域はローカルメモリ３２内に設けられた共有メモリ空間に配置される。

【００３３】図４および図５において，タスク管理部５１は，自ＣＰＵ上で動作するタスクを制御し，フォールトトレラント部５３からタスクの引き継ぎ依頼があれば，引き継ぐために待機していたタスクを起動し，停止依頼があればタスクを停止する。故障検出部５２は，自ＣＰＵまたは他のＣＰＵの故障を検出し，ＣＰＵ間通信部５５を介して他のＣＰＵに故障を通知する。フォールトトレラント部５３は，他のＣＰＵの故障を自ら検出するか，またはＣＰＵ間通信部５５を介して，他のＣＰＵの故障通知を受け取ると，再構成テーブル５４を参照して，自ＣＰＵがタスクを引き継ぐかどうかを判断する。ＣＰＵ間通信部５５は，各ＣＰＵ間でメッセージ通信を行うためのモジュールである。なお，図４に示すグローバルメモリ２４，または図５に示すローカルメモリ３２内の共有メモリ空間には，ＣＰＵ間の共通資源を管理するマルチＣＰＵ制御領域５６が設けられる。また，この例では，現在遷移位置テーブル５７も共有メモリ空間内に設けられている。

【００３４】図６に，本発明をコンピュータによって実現させるためのプログラムの処理フローチャートを示す。このフローチャートは，特にフォールトトレラント部５３を中心とした部分の処理の流れを示している。

【００３５】まず，システムにおける一つのＣＰＵの故障が検出された場合，再構成を行うために正常なＣＰＵ間で同期をとる（ステップＳ１１）。故障の検出は，例えば各ＣＰＵ間で定期的に生存確認のメッセージ交換を行うなど，従来から知られている方式を用いることができる。また，ＣＰＵ間の同期は，共有メモリ空間におけるマルチＣＰＵ制御領域５６またはＣＰＵ間通信部５５を用いて行うが，同期方法については，よく知られているので，ここでの詳しい説明を省略する。

【００３６】その後，再構成テーブル５４の先頭のタスクから順番に，現在遷移位置テーブル５７から得た現在遷移位置と，再構成テーブル５４における遷移情報から，着目するタスクが現在どのＣＰＵ上で動作しているのかを調べる（ステップＳ１２）。タスクが，既に動作していないタスクである場合，すなわち現在遷移位置テーブル５７から得た現在遷移位置が「－１」の場合，ステップＳ２１へ進む（ステップＳ１３）。また，現在遷移位置の示すＣＰＵが停止（故障）しているＣＰＵでない場合，ステップＳ２１へ進む（ステップＳ１４）。

【００３７】既に動作していないタスクではなく，また，そのタスクが割り当てられているＣＰＵが停止しているＣＰＵであった場合，再構成テーブル５４の遷移情報から次にそのタスクを動作させるＣＰＵを調べる（ステップＳ１５）。

【００３８】遷移情報に次のＣＰＵが存在しない場合，ステップＳ２３に進む（ステップＳ１６）。遷移情報に次のＣＰＵが存在するが，その次のＣＰＵも停止しているときには（ステップＳ１７），ステップＳ１５の処理へ戻り，再度，遷移情報から次にそのタスクを動作させるＣＰＵを調べる。

【００３９】また，次のＣＰＵが停止しているＣＰＵではない場合，そのタスクを引き継ぐＣＰＵが自ＣＰＵであるかどうかを調べる（ステップＳ１８）。自ＣＰＵである場合，ステップＳ１９へ進み，自ＣＰＵでない場合，ステップＳ２１へ進む。

【００４０】タスクを引き継ぐＣＰＵが自ＣＰＵである場合，現在遷移位置テーブル５７の現在遷移位置を更新し（ステップＳ１９），待機させていたタスクを起動する（ステップＳ２０）。

【００４１】ステップＳ２１では，再構成テーブル５４におけるすべてのタスクについて，以上の処理を行った

かどうかを調べ，すべてのタスクについて以上の処理が終わるまで，ステップＳ１２～Ｓ２０を繰り返す。再構成テーブル５４におけるすべてのタスクについて，以上の処理を行ったならば，他のＣＰＵの処理を待ち，同期をとる（ステップＳ２２）。正常のＣＰＵのすべてにおいて，それぞれタスク引き継ぎ処理が完了し，同期がとれたならば，新しいタスク構成により業務処理を継続する。

【００４２】また，ステップＳ１５の処理において，遷移情報から次にそのタスクを動作させるＣＰＵを調べ，次のＣＰＵが存在しない場合（ステップＳ１６），そのタスクの処理選択情報を調べる（ステップＳ２３）。処理選択情報がタスクであれば，現在遷移位置テーブル５７の現在遷移位置を「－１」とし，そのタスクがシステムからなくなったことを記し（ステップＳ２４），その後，ステップＳ２１へ進む。処理選択情報がタスクではなく，システム全体であれば，システム全体を停止させて処理を終了する（ステップＳ２５）。

【００４３】以上の実施の形態で説明したように，再構成テーブル５４を各ＣＰＵのローカルメモリ３２に配置し，現在遷移位置テーブル５７を共有メモリ空間に配置することによって，再構成テーブル５４へのアクセスの高速化と現在遷移位置の管理の容易化を実現することができ，好適なフォールトトレラント・システムを構築することができる。しかし，再構成テーブル５４と現在遷移位置テーブル５７とを，必ずしもローカルメモリ３２と共有メモリ空間とに分けて配置しなければならないわけではなく，例えばローカルメモリ３２または共有メモリ空間のいずれかにこれらのテーブルを共通に設けても，本発明を実施することができる。

【００４４】

【実施例】次に，入力されたデータを加工して出力する以下のようなタスクを持つシステムを例にして，本発明の実施例の処理動作を説明する。このシステムは４つのＣＰＵ１～ＣＰＵ４から構成されており，ＣＰＵ１とＣＰＵ３には，入力装置を制御できるコントローラが配置され，ＣＰＵ２とＣＰＵ４には出力装置を制御するコントローラが配置されているとする。

【００４５】本システムで稼働するタスクは，

・入力処理タスク
・出力処理タスク
・データ加工マスタタスク
・データ加工サブタスク

である。

【００４６】図７に，初期状態におけるタスクの構成例を示す。入力処理タスクはＣＰＵ１で，出力処理タスクはＣＰＵ２で稼働させる。データを加工する処理は分割して各ＣＰＵに分散させ，データ加工マスタタスクはＣＰＵ１に配置し，実際にデータを加工するデータ加工サブタスクは，すべてのＣＰＵにそれぞれ配置して並列に動作させる。

【００４７】また，データ加工マスタタスクは，ＣＰＵ２，ＣＰＵ３，ＣＰＵ４で待機させ，入力処理タスクはＣＰＵ３で，出力処理タスクはＣＰＵ４で，それぞれ待機させる。

【００４８】図８に，再構成テーブルおよび現在遷移位置テーブルの設定例を示す。再構成テーブル５４のタスクＩＤ，遷移情報，処理選択の情報を，図８に示すように設定して各ＣＰＵに配置する。再構成テーブル５４は，あらかじめプログラム中に組み込んでおいてもよい。入力処理タスクについては，ＣＰＵ１が故障した場合にはＣＰＵ３が引き継ぎ，引き継ぐＣＰＵがない場合にはシステム全体の処理を停止するように設定されている。

【００４９】出力処理タスクについては，ＣＰＵ２が故障した場合にはＣＰＵ４が引き継ぎ，これも引き継ぐＣＰＵがない場合にはシステム全体の処理を停止するように設定されている。データ加工マスタタスクは，稼働しているＣＰＵが故障すると，ＣＰＵ１，ＣＰＵ２，ＣＰＵ３，ＣＰＵ４の順番で順次使用可能なＣＰＵに処理が引き継がれ，使用可能なＣＰＵがなくなると，システム全体の処理を停止するように設定されている。データ加工サブタスク１は，ＣＰＵ１が故障しても他のＣＰＵは引き継がず，そのタスク処理を停止するように設定されている。データ加工サブタスク２，３，４も，それぞれ動作しているＣＰＵ２，ＣＰＵ３，ＣＰＵ４が故障すると，そのタスクは引き継がれない。

【００５０】現在遷移位置テーブル５７は，共有メモリ空間の領域に割り当て，各現在遷移位置（各タスクが現在どのＣＰＵ上で稼働しているのかを示すための遷移情報の系列中の位置）は，システムの初期化時にすべて「１」に設定される。

【００５１】今，ＣＰＵ２に故障が発生し，ＣＰＵ２を除いた構成でシステムを構成し直す場合，ＣＰＵ１，ＣＰＵ３，ＣＰＵ４は，再構成テーブル５４の遷移情報と現在遷移位置テーブル５７の現在遷移位置とを参照して，ＣＰＵ２上で稼働していたタスクを調べる。これにより，ＣＰＵ２上で稼働していたタスクは，「出力処理タスク」と「データ加工サブタスク２」であること，および，出力処理タスクはＣＰＵ４が引き継ぎ，「データ加工サブタスク２」は引き継ぐ必要がないことがわかる。

【００５２】遷移情報に従って，「データ加工サブタスク２」の現在遷移位置に「－１」を設定する。ＣＰＵ４は，「出力処理タスク」を引き継ぐために待機させていたタスクを起動し，「出力処理タスク」がＣＰＵ４に引き継がれたことを示すために，現在遷移位置テーブル５７における現在遷移位置を「２」に更新する。この状態のタスク構成は，図９に示すようになる。また，更新後の再構成テーブル５４および現在遷移位置テーブル５７

は，図１０に示すようになる。

【００５３】その後さらに，ＣＰＵ１に故障が発生して，ＣＰＵ１を除いてシステムを構成し直す場合，ＣＰＵ３，ＣＰＵ４は，現在遷移位置テーブル５７の現在遷移位置と再構成テーブル５４の遷移情報とを参照して，ＣＰＵ１で稼働していたタスクを調べ，ＣＰＵ１で稼働していたタスクが，「入力処理タスク」，「データ加工マスタタスク」，「データ加工サブタスク１」であること，および「入力処理タスク」はＣＰＵ３が引き継ぐこと，「データ加工サブタスク１」は引き継ぐ必要がないことがわかる。また，「データ加工マスタタスク」はＣＰＵ２が引き継ぐことがわかるが，ＣＰＵ２は既に故障しているので，さらに次の遷移情報を調べて，ＣＰＵ３が引き継ぐことを認識する。なお，ＣＰＵ２が既に故障しているかどうかは，共有メモリ空間のマルチＣＰＵ制御領域５６に記されているので，それによって認識する。

【００５４】ＣＰＵ３は，遷移情報から，「入力処理タスク」および「データ加工マスタタスク」を引き継ぐことを認識すると，これらを引き継ぐために待機させていたタスクをそれぞれ起動して，「入力処理タスク」の現在遷移位置を「２」に，「データ加工マスタタスク」の現在遷移位置を「３」に更新する。また，「データ加工サブタスク１」の現在遷移位置に「－１」を設定する。この状態のタスク構成は，図１１に示すようになる。また，更新後の再構成テーブル５４および現在遷移位置テーブル５７は，図１２に示すようになる。

【００５５】

【発明の効果】以上説明したように，本発明によれば，マルチＣＰＵシステムにおいて，故障したＣＰＵ上のタスクをどのＣＰＵによって引き継ぐのかを，再構成テーブルにＣＰＵの番号を羅列して記すことにより，容易に引き継ぎの順序を指定することができ，さらに，現在どのＣＰＵ上でタスクが実行されているのかを示すことで，次に引き継ぐＣＰＵを簡単に調べることができる。

【００５６】これにより，遷移情報ですべてのＣＰＵを指定することによって，正常なＣＰＵが最後の一つになるまで引き継ぐような指定や，特定のいくつかのＣＰＵだけが引き継ぎを行うような指定を，簡単に設定することができる。

【００５７】さらに，タスクごとの処理選択情報を再構成テーブル中に持たせることにより，引き継ぐＣＰＵがなくなった場合にそのタスクだけを停止させたり，またはシステム全体を停止させたりするような指定も，簡単に設定することができる。

【００５８】また，あらかじめ引き継ぐＣＰＵを設定しておくため，タスクに必要なハードウェアを備えるＣＰＵにだけ確実に引き継ぎを行わせることも可能になる。

【図面の簡単な説明】

【図１】本発明の概要を示すブロック構成図である。

【図２】本発明の処理動作の流れの例を示す図である。

【図３】マルチＣＰＵシステムの接続形態の例を示す図である。

【図４】疎結合型システムにおけるブロック構成例を示す図である。

【図５】密結合型システムにおけるブロック構成例を示す図である。

【図６】本発明をコンピュータによって実現させるためのプログラムの処理フローチャートである。

【図７】本発明の実施例の初期状態におけるタスクの構成例を示す図である。

【図８】本発明の実施例の再構成テーブルおよび現在遷移位置テーブルの設定例を示す図である。

【図９】システムの各ＣＰＵのタスクの状態を説明するための図である。

【図１０】更新後の再構成テーブルおよび現在遷移位置テーブルの例を示す図である。

【図１１】システムの各ＣＰＵのタスクの状態を説明するための図である。

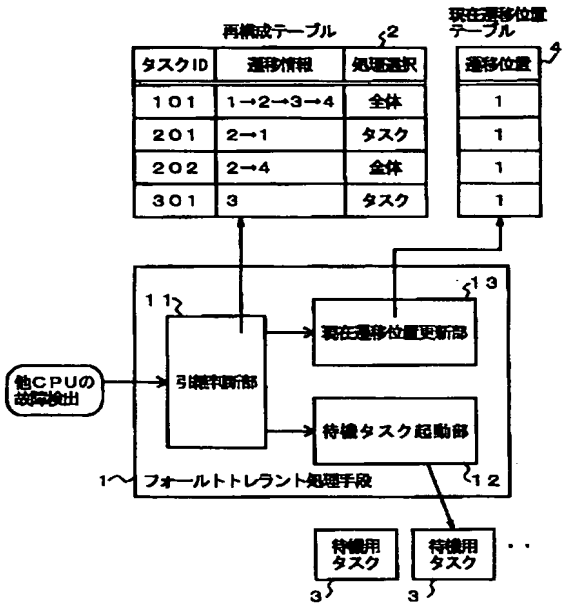
【図１２】更新後の再構成テーブルおよび現在遷移位置テーブルの例を示す図である。

【符号の説明】

１　フォールトトレラント処理手段
１１　引継判断部
１２　待機タスク起動部
１３　現在遷移位置更新部
２　再構成テーブル
３　待機用タスク
４　現在遷移位置テーブル

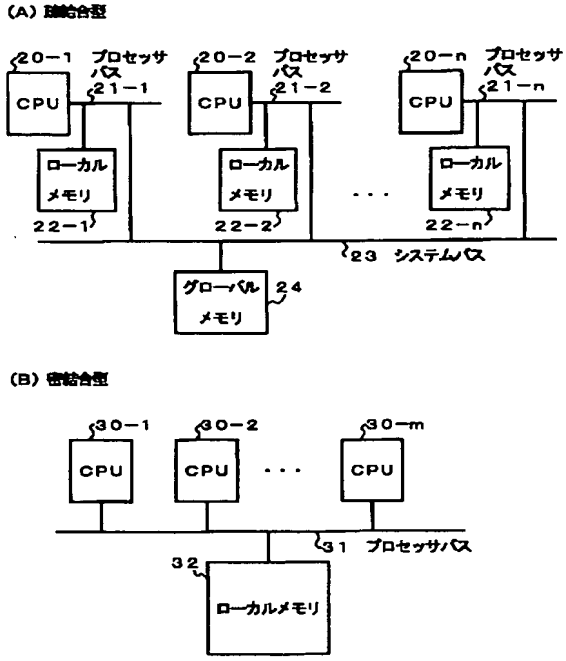
【図１】

本発明のブロック構成図



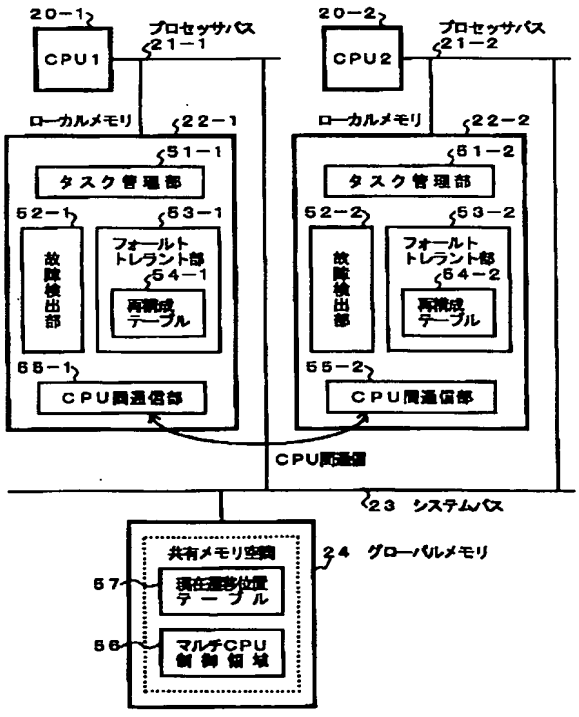
【図３】

接続形態の例

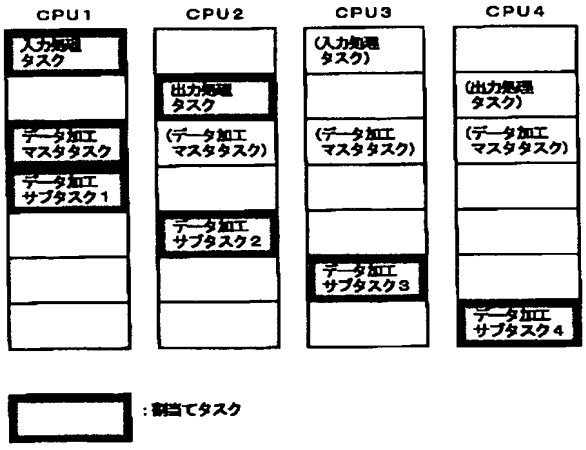


【図４】

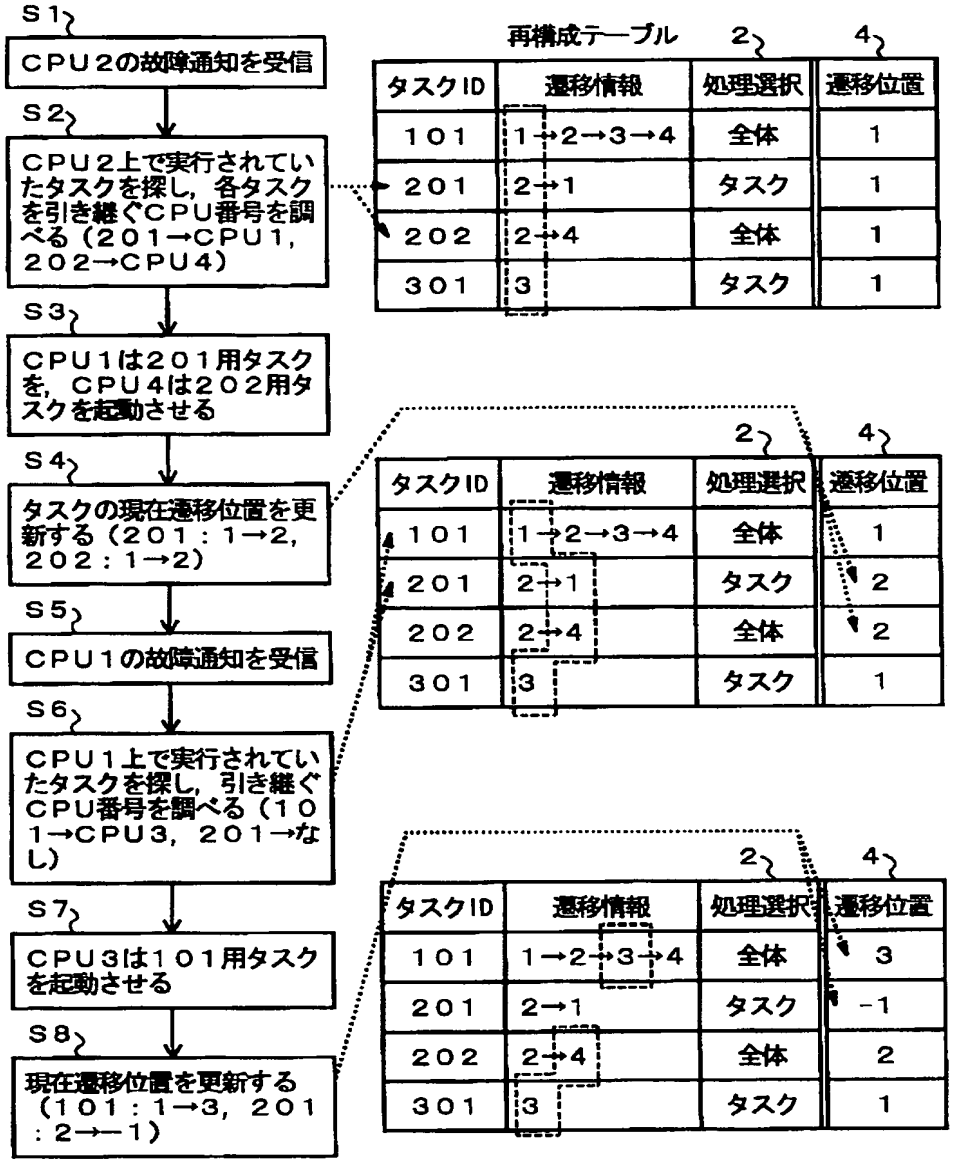
疎結合型システムにおけるブロック構成例



【図７】

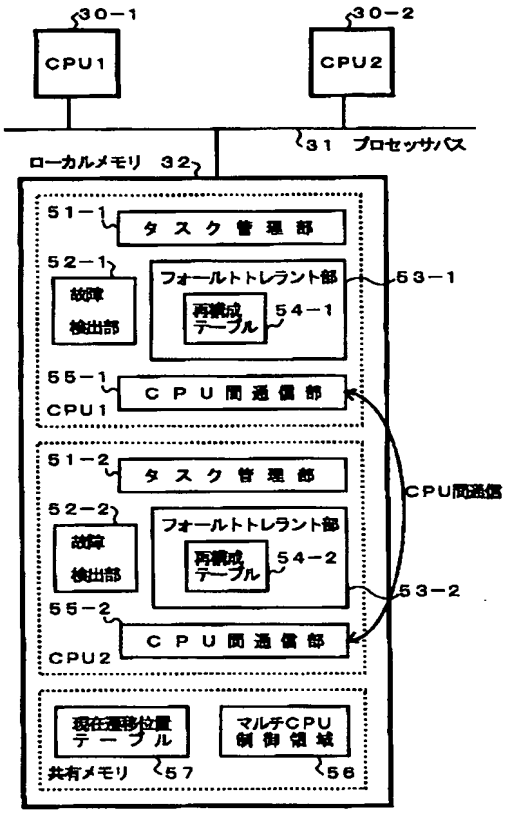


【図２】


S1
ＣＰＵ２の故障通知を受信
S2
ＣＰＵ２上で実行されていたタスクを探し，各タスクを引き継ぐＣＰＵ番号を調べる（２０１→ＣＰＵ１，２０２→ＣＰＵ４）
S3
ＣＰＵ１は２０１用タスクを，ＣＰＵ４は２０２用タスクを起動させる
S4
タスクの現在遷移位置を更新する（２０１：１→２，２０２：１→２）
S5
ＣＰＵ１の故障通知を受信
S6
ＣＰＵ１上で実行されていたタスクを探し，引き継ぐＣＰＵ番号を調べる（１０１→ＣＰＵ３，２０１→なし）
S7
ＣＰＵ３は１０１用タスクを起動させる
S8
現在遷移位置を更新する（１０１：１→３，２０１：２→－１）
再構成テーブル
2
4
タスクID
遷移情報
処理選択
遷移位置
１０１
１→２→３→４
全体
1
２０１
２→１
タスク
1
２０２
２→４
全体
1
３０１
３
タスク
1
2
4
タスクID
遷移情報
処理選択
遷移位置
１０１
１→２→３→４
全体
1
２０１
２→１
タスク
２
２０２
２→４
全体
２
３０１
３
タスク
1
2
4
タスクID
遷移情報
処理選択
遷移位置
１０１
１→２→３→４
全体
３
２０１
２→１
タスク
－１
２０２
２→４
全体
2
３０１
３
タスク
1

【図５】

密結合型システムにおけるブロック構成例



【図８】

テーブルの設定例

再構成テーブル 54

タスク	遷移情報	処理選択
入力処理タスク	1→3	全体
出力処理タスク	2→4	全体
データ加工マスタタスク	1→2→3→4	全体
データ加工サブタスク１	1	タスク
データ加工サブタスク２	2	タスク
データ加工サブタスク３	3	タスク
データ加工サブタスク４	4	タスク

現在遷移位置テーブル 57

遷移位置
1
1
1
1
1
1
1

【図１０】

再構成テーブルおよび現在遷移位置テーブルの例

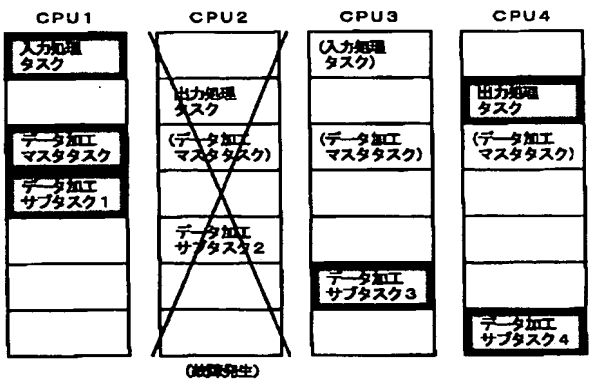
再構成テーブル 54

タスク	遷移情報	処理選択
入力処理タスク	1→3	全体
出力処理タスク	2→4	全体
データ加工マスタタスク	1→2→3→4	全体
データ加工サブタスク１	1	タスク
データ加工サブタスク２	2	タスク
データ加工サブタスク３	3	タスク
データ加工サブタスク４	4	タスク

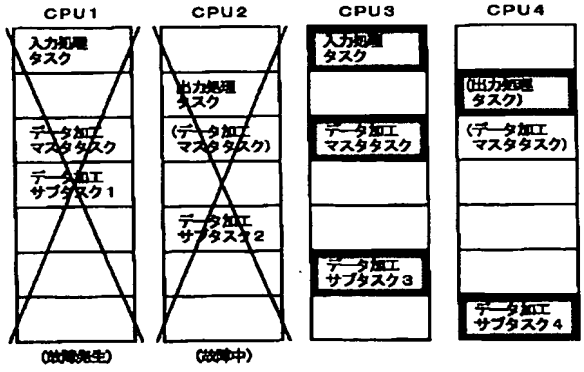
現在遷移位置テーブル 57

遷移位置
1
2
1
1
-1
1
1

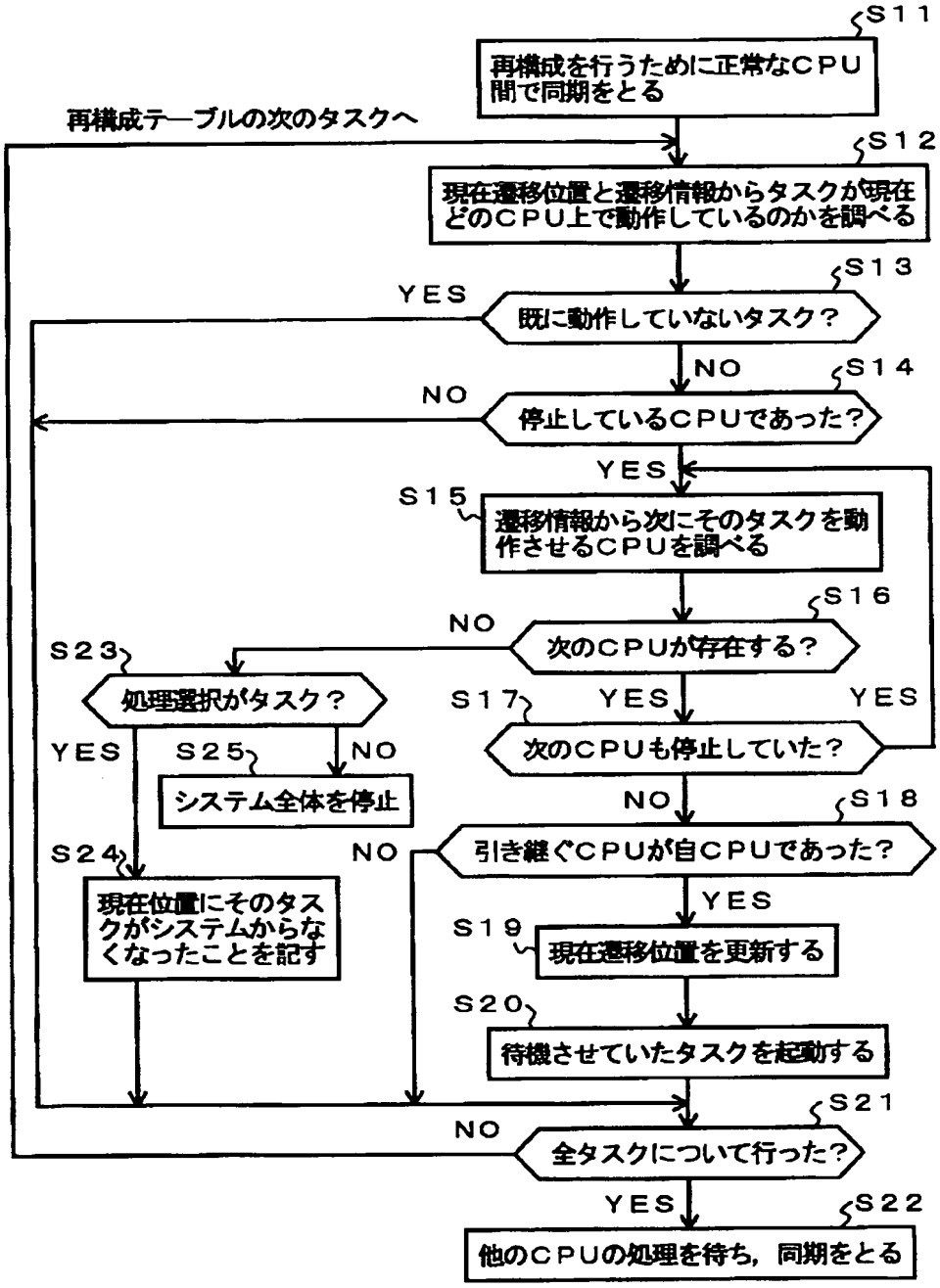
【図９】



【図１１】



【図６】


S11
再構成を行うために正常なＣＰＵ間で同期をとる
再構成テーブルの次のタスクへ
S12
現在遷移位置と遷移情報からタスクが現在どのＣＰＵ上で動作しているのかを調べる
S13
YES
既に動作していないタスク？
NO
S14
NO
停止しているＣＰＵであった？
YES
S15
遷移情報から次にそのタスクを動作させるＣＰＵを調べる
S16
NO
次のＣＰＵが存在する？
S23
処理選択がタスク？
S17
YES
YES
次のＣＰＵも停止していた？
YES
S25
NO
システム全体を停止
NO
S18
S24
NO
引き継ぐＣＰＵが自ＣＰＵであった？
現在位置にそのタスクがシステムからなくなったことを記す
YES
S19
現在遷移位置を更新する
S20
待機させていたタスクを起動する
S21
NO
全タスクについて行った？
YES
S22
他のＣＰＵの処理を待ち，同期をとる

【図１２】

再構成テーブルおよび現在遷移位置テーブルの例

再構成テーブル　５４

タスク	遷移情報	処理選択
入力処理タスク	1→3	全体
出力処理タスク	2→4	全体
データ加工マスタタスク	1→2→3→4	全体
データ加工サブタスク１	1	タスク
データ加工サブタスク２	2	タスク
データ加工サブタスク３	3	タスク
データ加工サブタスク４	4	タスク

現在遷移位置テーブル　５７

遷移位置
2
2
3
-1
-1
1
1

フロントページの続き

(72)発明者　岡本　二朗
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

Ｆターム(参考)　5B034 BB11 CC01
5B045 BB02 BB12 GG06 JJ09 JJ13 JJ44
5B098 AA10 GA04 JJ00